RAL.GS.77
RAL.GS.77

MISSION OPERATIONS COST REJ3UCTION BY SOFTWARE INHERITANCE

John A. Rohr
Keyur C. Patel
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California USA
John.A.Rohr@Jpl.Nasa.Gov

Since the **initiation** of deep-space exploration in the 1960's by spacecraft built by the Jet Propulsion Laboratory (JPL), almost all JPL mission operations software has been written for and operated on mainframe computers. In the late **1980's, a phenomenon began which would have a profound effect** on JPL mission operations software: the emergence of open systems in which an operating system and the applications programs it controlled could run with only minimal changes on computer systems made by different manufacturers.

At the time, mainframe computers were becoming increasing expensive to operate when compared with workstations. In addition, software written for mainframe computers was designed to operate with a specific mainframe computer operating system. Converting such software to another mainframe system required considerable resources just to convert to a different operating system, even if the functional program code were unchanged. Most workstations, then as now, support the UNIX operating system. Therefore, moving software which was designed to run under the UNIX operating system from one workstation to another would require only minimal changes to the software due to minor differences in the implementation of the UNIX operating system on a particular vendor's hardware platform. This open system commonality was appealing to the designers and implementers of JPL mission operations software.

Furthermore, in the late 1980's JPL began to investigate the concept of building a set of operations software which could be used by multiple missions and would require only a minimal effort for adaptation from one mission to another. The concept of multi mission software was appealing to management, because considerable savings could be realized by having a core set of software which could be mostly reused from mission to mission with only minor adaptation required.

The combination of the availability and capability of workstations and the concept of multimission software led to the JPL Multimission Software Transition Project (MSTP). The project official] y began in 1993 with an expected completion by the end of 1996. The project has two objectives. The first is to convert mission operations software from mainframe computers to UNIX workstations. The second is to convert the programs in such a way that the resulting software will be multimission in nature and will require only minimum adaptation from mission to mission.

The original scope of MSTP was the conversion of all JPL mission operations software which was usable by multiple missions, and all JPL mission operations software which operated on UNISYS 1100 mainframe computers and was being used by one or more missions which would continue beyond the time that MSTP would be completed. Accomplishment of this objective would mean that the UNISYS 1100 mainframe computers could be decommissioned after completion of MSTP, since all software which operated on these machines would be converted to workstations by MSTP. In addition, some software which was originally designed for a single mission or a limited set of missions was to be modified by MSTP to become multimission software which could be used by several missions with only minimal adaptation.

MSTP was originally scheduled to begin in **1991.** Due to a situation in the NASA budgeting process, MSTP did not actually start until 1993. This two-year delay resulted in several modifications to the original MSTP plan.

First, at the time of the initiation of MSTP a national effort was in progress to reduce the amount of money spent by the federal government. JPL is operated by the California Institute of Technology under a contract with the National Aeronautics and Space Administration (NASA) which was expected to participate in this budget-reduction process. As a result, there was a strong desire on the part of NASA to reduce the MSTP budget (as well as other NASA budgets). To help NASA meet its budget-reduction goals, the MSTP budget was reviewed and reductions made where feasible. Some of the reductions resulted from more accurate cost estimates for specific tasks in MST]'. Other reductions resulted from elimination of software which did not need conversion because it would be replaced soon by other software being written, Additional reductions were obtained where software conversion to workstations was already being funded by other sources.

The second modification to MSTP due to the delayed start was the deletion of conversion of mission operations software unique to Galileo. The Galileo mission, scheduled to complete its planned mission by 1998, would be almost complete by the revised completion date of MST]'. Since much of the Galileo software was designed specifically for that mission and operated on UNISYS mainframe computers, the main reason for MSTP to convert Galileo mission operations software was to move it from the UNISYS mainframes to UNIX workstation so that the UNISYS mainframes could be decommissioned. If MSTP had been completed in 1994 as originally planned, Galileo would have had four years to run its mission operations software on UNIX workstations. With the delay of MSTP, the Galileo mission would have only two years to run its mission operations software on UNIX workstations. This would be of marginal benefit since the cost of converting mission operations software unique to Galileo was planned to be a significant part of the MST]' budget. It was determined that a considerable savings would result if Galileo continued to run its mission operations software on the UNISYS mainframes through the duration of the planned mission rather than switching to UNIX workstations. Thus, conversion of mission operations software unique to Galileo was deleted from MSTP to save money.

The third modification to MSTP due to the delayed start was the incorporation of new technology which had evolved between the original start date and the delayed start date. This will be illustrated in the discussion of the replacement of the COMSIM program below.

Thus, between the original starting date of 1991 and the revised starting date in 1993 when MSTP actually began, the cost and scope of the project were reduced significantly,

Two specific examples illustrate how MSTP will reduce the overall cost of mission operations software: the conversion of the Sequence Translator program (SEQTRAN) and the replacement of the COMSIM data system simulator program. Two completely different conversion methods were used for these two programs. SEQTRAN was converted by translating UNISYS mainframe assembly-language programs to C-language programs running under the UNIX operating system. The COMSIM program was completely replaced by an adaptation of a multimission simulator program. The COMSIM replacement is an example of use of new technology. The conversion of each of these programs will be discussed to illustrate how different methods of conversion can be used effectively to achieve cost reduction.

The SEQTRAN program is used to translate spacecraft commands in the form of mnemonic instructions with parameters into the binary data required onboard the spacecraft to drive the command and sequencing interpreter. SEQTRAN also includes other functions such as management of onboard memory and specialized output file formatting.

SEQTRAN is one of several programs used in JPL mission operations to combine requests for science activities and engineering activities onboard the spacecraft into a sequence of events to be executed onboard the spacecraft. The SEQ. GEN program collects the requests and produces a file which is input to SEQTRAN. This file contains a list of parameterized commands to be executed onboard the spacecraft. SEQTRAN accepts the input file and produces an output file in one of three formats, depending on mission requirements. These output files are sent to the Command System which processes the files to transform them into files which are sent to the Deep Space Network for transmission to the spacecraft.

Two changes were made to SEQTRAN in addition to conversion to operation on a UNIX workstation, which will reduce the cost of future mission operations, First, all SEQTRAN versions were combined into one. Previously, a slightly different version of SEQTRAN was generated for each different mission. All versions have now been combined and an internal routine configures the program for the mission being supported. This means that only one version of the program will need to be maintained rather than the multiple versions which were required previously. Second, command translation is accomplished using a common library provided for all programs which need to translate commands into bits. This function previously required separate coding of the translation in SEQTRAN as well as other mission operations programs.

Since its inception, SEQTRAN has been programmed and run on UNISYS **1100** mainframes. Because SEQTRAN was based code inherited from earlier work, it is written entirely in UNISYS 1100 assembly language. MST]' will convert SEQTRAN from UNISYS 1100 assembly language to the C language running under the UNIX operating system on Sun and Hewlett-Packard workstations. To minimize cost and schedule, SEQTRAN was converted to C by emulating the assembly-language code rather than rewriting each function or the entire program completely. One other reason for this approach was that the programmers working on the conversion were not familiar with SEQTRAN before beginning the conversion and thus did not understand the details and algorithms of the assembly-language implementation.

The result of the SEQTRAN conversion is a single C-language program which supports multiple missions, The converted program has been tested and is currently supporting four JPL missions. The use of the common command translator will minimize the effort required to use SEQTRAN on new missions and the unified version will reduce maintenance costs in the future.

The COMSIM program is a data system simulator for the Voyager Project. COMSI M provides a detailed simulation of the onboard Voyager control computer and a functional simulation of the Voyager spacecraft environment in which the control computer operates. COMSIM runs on UNISYS 1100 mainframe completers.

The Voyager Project has utilized the COMSIM simulator since before launch to develop flight software and test sequences which are to be sent to the spacecraft. Voyager personnel have also made extensive use of COMSIM for anomaly invest igation since the launch of the Voyager spacecraft.

Several other JPL missions have also utilized data system simulators to develop flight software, to test sequences before sending them to the spacecraft, and to investigate anomalies which occur in flight. The simulators model the onboard control computer hardware and simulate the surrounding environment so that actual flight software can be run instruction-by-instruction in the simulator. The use of faster onboard processors by recent missions generally means that a simulation of the onboard data system will run much slower than real-time.

COMSIM was originally written in a combination of several languages. FORTRAN was the main language used, but assembly language was used for simulator code which was inherited from a Voyager simulator written before COMSIM. Also, some stand- alone utilities were developed for COMSIM using languages other than FORTRAN, The original plan for the conversion of COMSIM was to convert the entire program to a UNIX workstation using standard FORTRAN. Since a common language was desired for all the code in the converted program and since the majority of the COMSIM code was written in FORTRAN which is supported on UNIX workstations, FORTRAN was chosen over C as the language for the converted COMSIM program.

Since the start of MSTP was delayed, the conversion method for COMSIM was reviewed before beginning the work. A research effort had been initiated several years ago to investigate the use of fast, multiple-processor UNIX workstations to build a data system simulator which would run at least as fast as real time. Following several years of research, a multimission high-speed simulator framework has been developed to provide a basis for building data system simulators for several missions. One version of the high-speed simulator has been built for the Galileo project and another is being built for the Cassini project,

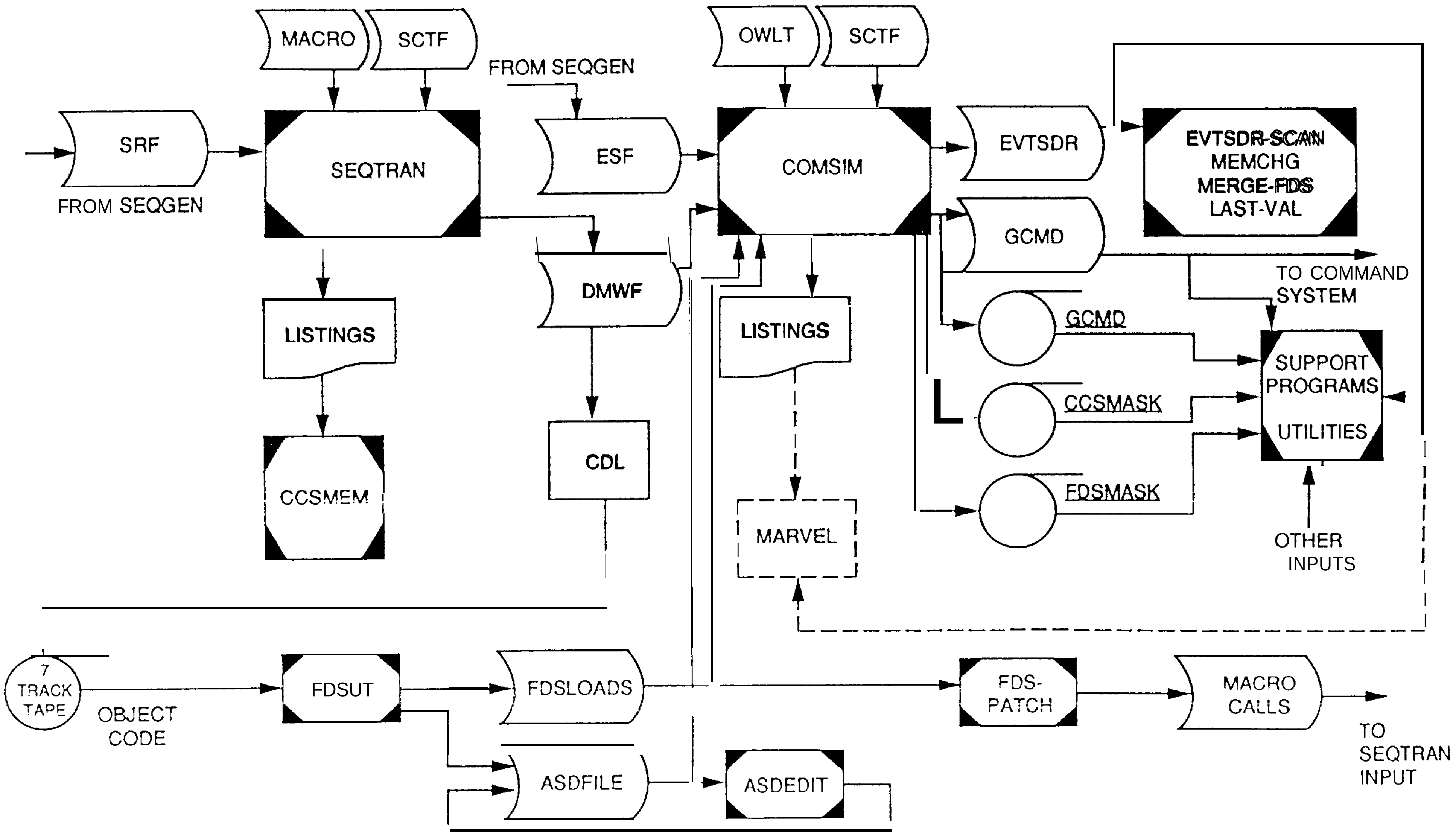
Because of the success of the high-speed simulator research effort, the implementation of a high-speed simulator for Galileo, and the planned implementation of a high-speed simulator for Cassini, it was decided to end the effort to convert COMSIM to UNIX workstations and build an adaptation of the high-speed simulator instead. This would not only save development funds, but it would also give the Voyager project a data system simulator which is much more modern and capable than COMSIM as well as being maintainable in a multimission mode. The decision to use an adaptation of the high-speed simulator rather than converting COMSIM is a win-win situation in which the MSTP reduces costs and the Voyager project receives a simulator with more speed and capability than would be obtained from a conversion of COMSIM.

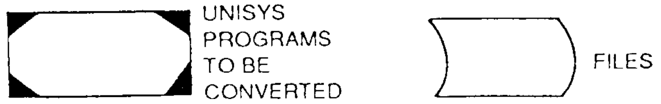
This was not a straightforward decision, however, because at the time the decision had to be made, the high-speed simulator had only been used for one project (Galileo), and it would not be completely finished until after the development for Voyager was to have begun. Thus there was not any completely-developed version of the high-speed simulator for any real mission. Only prototypes had been completed when the decision had to be made as to how to proceed for Voyager.

Nevertheless, the decision was made to use the high-speed simulator adaptation and the adaptation was initiated.

At the present time, the adaptation of the high-speed simulator for Voyager is still in progress. Early versions have been demonstrated and are being used by Voyager personnel for training. It is expected that the high-speed simulator will be an even more effective tool for the Voyager project than COMSIM.

The result of the MSTP project is multimission software which runs on open-system workstations. The use of multimission software will result in software which is cheaper to develop and maintain and which can be transferred to new, upgraded computer systems with minimal effort. The final result is reduced future mission operations cost at JPL.





VOYAGER MISSION OPERATIONS SOFTWARE FLOW DIAGRAM